kNN 案例

MNIST 训练集由来自 250 个人手写的数字构成，其中 50%是高中学生,50%来自人口普查局的工作人员. 测试集也是同样比例的手写数字数据

```
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification_report

from sklearn.utils import shuffle

from sklearn.datasets import fetch_mldata

from sklearn.cross_validation import train_test_split

# 数据集 MNIST
mnist = fetch_mldata("MNIST original")

mnist.data,mnist.target = shuffle(mnist.data,mnist.target)

mnist.data = mnist.data[:1000]

mnist.target = mnist.target[:1000]

X_train,X_test,y_train,y_test =
     train_test_split(mnist.data,mnist.target,
                     test_size=0.8,random_state=0)

# 训练模型并测试
model = KNeighborsClassifier(3)

model.fit(X_train,y_train)

y_predicted = model.predict(X_test)

print classification_report(y_test,y_predicted)
```

```
           precision    recall  f1-score   support

      0.0       0.75      0.89      0.81        81
      1.0       0.60      0.99      0.75        83
      2.0       0.94      0.64      0.76        92
      3.0       0.76      0.70      0.73        87
      4.0       0.71      0.57      0.63        74
      5.0       0.76      0.78      0.77        54
      6.0       0.87      0.90      0.88        89
```

```
       7.0      0.86     0.79     0.83      87
       8.0      0.85     0.56     0.67      72
       9.0      0.65     0.73     0.69      81

avg / total      0.78     0.76     0.76      800

%timeit clf.fit(X_train,y_train)
100 loops, best of 3: 5.51 ms per loop

%timeit clf.predict(X_test)
1 loop, best of 3: 535 ms per loop
```

算法在训练阶段仅仅拷贝数据。预测速度与训练阶段使用的样本数目和构成数据集的特征数目有关。其他算法中，预测速度都独立于所使用的数据集的训练样本数。
总之，kNN 对小数据集非常好，但处理大数据集时不宜使用。

```
# 再看一个例子,介绍人工生成分类数据集及分层划分训练/测试集

from sklearn.datasets import make_classification

from sklearn.cross_validation import StratifiedShuffleSplit

# 生成一个数据集(100x4)及类标签(0/1)
X,y = make_classification(n_features=4)

ds = np.column_stack([X,y])

strat_split = StratifiedShuffleSplit(ds[:,-1],test_size=0.2,
                                      n_iter=1)

for train_idx,test_idx in strat_split:
    X_train = ds[train_idx,:-1]
    y_train = ds[train_idx,-1]
    X_test = ds[test_idx,:-1]
    y_test = ds[test_idx,-1]

model = KNeighborsClassifier(n_neighbors=3)

model.fit(X_train,y_train)

# 对训练集预测评估
y_predicted = model.predict(X_train)
```

```
print classification_report(y_train,y_predicted)
           precision   recall  f1-score  support

       0.0      1.00     0.95     0.97      40
       1.0      0.95     1.00     0.98      40

avg / total      0.98     0.97     0.97      80

# 对测试集预测
y_predicted = model.predict(X_test)

accuracy_score(y_test,y_predicted)
0.95

confusion_matrix(y_test,y_predicted)
array([[ 9,  1],
       [ 0, 10]])

print classification_report(y_test,y_predicted)
           precision   recall  f1-score  support

       0.0      1.00     0.90     0.95      10
       1.0      0.91     1.00     0.95      10

avg / total      0.95     0.95     0.95      20
```